HCTL Open Science and Technology Letters (HCTL Open STL)
Edition on Reconfigurable Computing - Embedded, FPGA based, VLSI and ASIC Designs, June 2013
e-ISSN: 2321-6980, ISBN (Print): 978-1-62776-963-1

A Novel Logic Function Acquisition Methodology

Chin Kok Loon*
klchin@altera.com

Abstract

The aim of this paper is to present a viable, alternative methodology to acquire function from custom circuit. Conventional function acquisition methodology is done using behavioural simulator or spice simulator to extract the functional state of circuit design either in Verilog or SPICE format. The research for an alternative methodology is therefore, not to replace the stated conventional method that is not working but rather as a mean to improve the efficiency of obtaining the required result, while at the same time, being a more cost effective solution.

Introduction

Moore's Law stated that the number of transistors that can be inexpensively placed on an integrated circuit is increasing exponentially, doubling approximately every two years [3]. But with the exponential increase in processing

*Altera

e-ISSN: 2321-6980, ISBN (Print): 978-1-62776-963-1

power comes the challenge that the size of data sets to be analyzed increases as well. Such increase in data sets inevitably leads to an increase in the time complexity of solving a problem. The time complexity of solving a problem by an algorithm can be explained by the big O notation, which is used to describe how the size of input data affects an algorithm's usage of computational resources [4]. The $O(f(x))$ denotes a collection of function $g(x)$ that exhibit a growth that is limited to that of $f(x)$ in some respect. The traditional notation for stating that $g(x)$ belongs to this collection is: $g(x) = O(f(x))$. For example, $g(x) = 2x^2$, $g(x) = 5x^2$, and $g(x) = 10x^2 + 10000$ have $O(x^2)$ because as x grows large, the x^2 term will dominate. As we shall see later, using conventional method such as Verilog and SPICE to perform function acquisition of a standard cell circuit with n inputs can be expressed as: $t(n) = x * n^2$, where x equals to the number of steps required to be done in order to produce logic output for single input combination. The alternative approach described in this paper will attempt to reduce the time complexity of function acquisition while at the same time does not incur additional research and development investment costs.

Conventional Logic Function Acquisition Flow

The prerequisite for standard cell timing model, the logic function of each circuit must be known. The conventional industry standard flow for function acquisition of a standard cell is to run either Verilog or SPICE simulation. For typical in house timing model generation with few hundreds of standard cells, a few hundred simulation iterations required to cover all the required process, voltage and temperature. A conventional Verilog or SPICE based simulation flow is shown in figure 1.

The conventional simulation flow shown in figure 1 has time complexity of 5. This is because the flow is shown to be consisted of 5 steps process. For the sake of argument, assumption is made here that these are the steps that cannot be avoided in both conventional Verilog and SPICE simulation flow. Although variant flows that included more steps are definite possible, the steps described in figure 1 are crucial steps that cannot be avoided in both Verilog and SPICE simulations. A precise time complexity is obtained when the time complexity of the dominant step of the flow is taken into account. For the simulation flow in figure 1, the step to prepare input source value for each input vector is the dominant one, as it can be represented by equation $n * p$ where,where n is the series of all possible input combination and p is the number of input vector of a CHLE circuit. For the purpose of function acquisition,

e-ISSN: 2321-6980, ISBN (Print): 978-1-62776-963-1

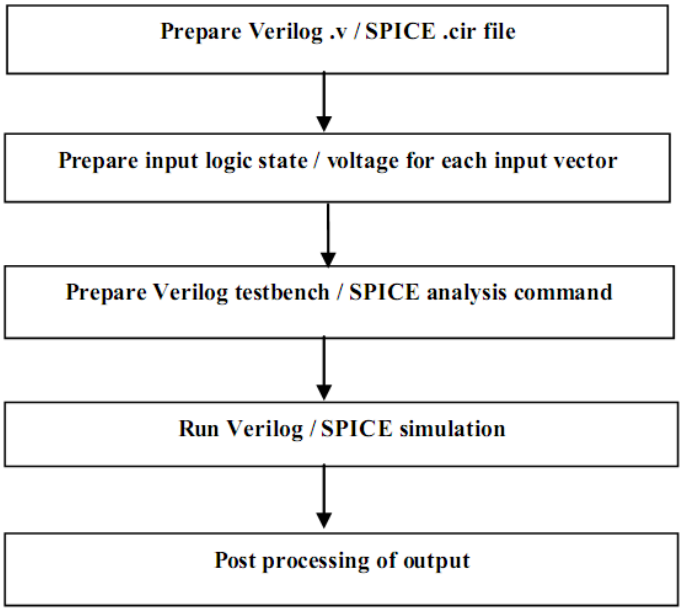

Figure 1: *Conventional Verilog or SPICE simulation flow*

input vector of a digital logic circuit can be represented by 2 states, 0 and 1. Therefore, $n = p^2$, and substituting this into equation $n * p$ produces $p^2 * p = p^3$. Therefore, time complexity of conventional Verilog and SPICE simulation flow for the purpose of function acquisition has time complexity of $4 + p^3$. In the event of worst case number of input vector or port of a large complex standard cell is 10, the time complexity is $4 + 10^3 = 1004$. With consideration of typically has 200 to 300 standard cells, the worst case total time complexity ranges from $200 * 1004 = 200800$ to $200 * 1004 = 301200$.

This large time complexity will make the task of setting up Verilog or Spice simulations for standard cell function acquisition purposes extremely tedious and time consuming. At the same time, the possibility of having missing input vectors and incorrect input vectors setup in simulation cannot be excluded, which eventually lead to simulation reruns of these customized standard cells. These will add delay to the design cycle time of obtaining customized standard cells function, thus affecting timing model generation. Although delays can be partially migated through additional investment in Verilog and SPICE licenses, this means increasing R&D costs into the project. At time same time, it does not reduce the amount of time complexity that engineer will face using conventional

e-ISSN: 2321-6980, ISBN (Print): 978-1-62776-963-1

function acquisition flow.

Logic Function Acquisition Methodology using DSPF Approach

In order to solve current problems explained previously, a cost effective approach need to be developed as an alternative to the Verilog and Spice based simulation approach. Not only this approach will need to be cost effective, it will need to be proven to be capable of handling both combinational and sequential logic found in customized standard cells. It will need to be easier to setup compared with Verilog and Spice based approach. It will need to be a method where result will be free from human errors. It will need to be implemented in a way that users will not need to manually engage in tasks involving evaluation of transistors logic and tracing of circuit. The DSPF approach begins by performing transistor level parasitic extraction on a circuit layout to obtain a DSPF file. A DSPF file contains information such as the device parameters, connectivity between devices, the type and name of devices. This information from DSPF can then be supplied to an algorithm that uses such data to determine the function of a customized standard cell. A port information file is created to provide information regarding input and output ports. These are inputs to the DSPF function acquisition algorithm. The function of a customized standard cell is then acquired from the main algorithm. This algorithm required single implementation, and will work with both combinational and sequential logic circuit, without the hassle of performing individual Verilog and spice setup and simulation for each circuit. Thus, this approach has a constant time complexity of 3 from user point of view for each customized standard cell circuit - obtaining DSPF from layout, preparing port file, and passing inputs to main algorithm. One crucial advantage seen in this flow is the elimination of input vector preparation which is a requirement of both Verilog and Spice approach. As shown previously, the time complexity of conventional Verilog and SPICE simulation flow has time complexity of $4 + p^3$ due to the dominant factor from input vector preparation. Comparing both time complexities, the equation $3 < 4 + p^3$ holds true. Therefore, DSPF approach is shown to be more efficient than conventional methods. In the following section, the algorithm to perform this function acquisition method will be discussed.

DSPF Function Acquisition Algorithm

HCTL Open Science and Technology Letters (HCTL Open STL)
Edition on Reconfigurable Computing - Embedded, FPGA based, VLSI and ASIC Designs, June 2013
e-ISSN: 2321-6980, ISBN (Print): 978-1-62776-963-1

DSPF Data Storage

The algorithm is consisted of a group of functions working together. The first part of the algorithm is called DSPF data storage, where information is obtained from a DSPF file. The transistor instances can be obtained from the instance section area of the DSPF file. Figure 2 showed an example portion of DSPF instance section.

The net name can be obtained from each $*|NET$ section of a DSPF file.

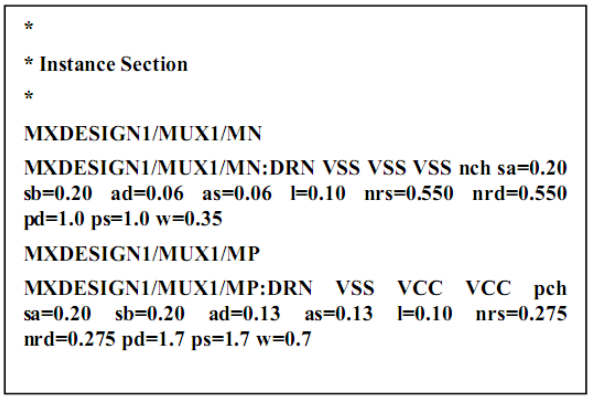

Figure 2: *The instance section of a DSPF file*

From this section, we also determined the terminals, identified by $*|I$, connected to this net. Refer to figure 3 for an example.

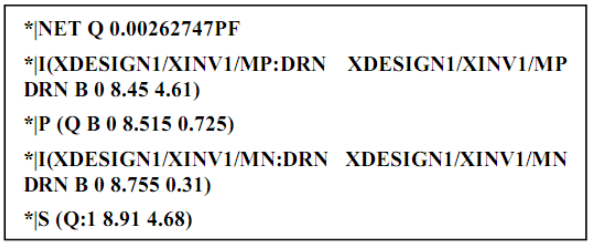

Figure 3: *The net section of a DSPF file*

Initializing Z State and Current Cycle

The second part of the function acquisition algorithm involved initializing all nets and non power and non ground connected terminals to Z states. This

e-ISSN: 2321-6980, ISBN (Print): 978-1-62776-963-1

is due to the fact that in the beginning, nets and terminals (non power and ground connected) have no signal values. Also, a property called current cycle need to be set to 0 for nets and terminals (non power and ground connected). This current cycle property has the form of 0, 1 and 2. The cycle property can be thought of as a count whenever a new signal arrives. 0 is used to represent no signal available. 0 is switched to 1 when the 1^{st} signal arrives to a net and connected terminal. The transition of 1 to 2 occurred when the signal changes value again. The cycle property of 2 is used for sequential logic determination, where rising and falling clock edge need to be evaluated as two input combination vectors-before and after the active edge. During the `before` period, arriving signals will have cycle property of 1. During the `after` period, arriving signals and stable signals (no change in state) will be set cycle property of 2. For power and ground nets and terminals connected directly to them, the cycle property will always be 2, as the value persisted throughout `before` and `after` active edge. Figure 4 shows the point being discussed, while figure 3 explains where we can obtain the net names and terminals in order to create a list that store their states and cycle properties. As for current implementation, the customized standard cell functions is assumed to be stabilized within single clock cycle.

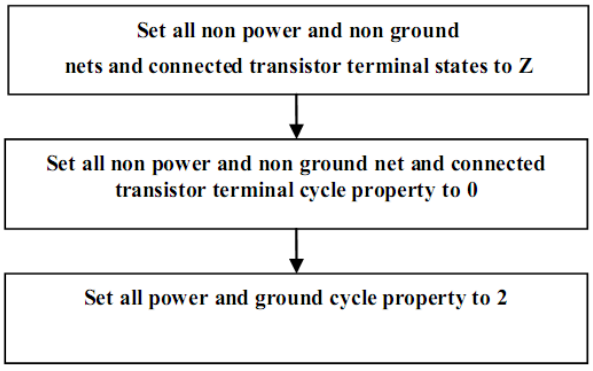

Figure 4: *Initializing Z state and current cycle property*

Generating Input Vectors

The next step is the generation of input vectors by using information from the port file. The structure and data of an example D Flip Flop port file is shown in figure 5.

e-ISSN: 2321-6980, ISBN (Print): 978-1-62776-963-1

From the port file, three kinds of data are supplied to the function ac-

D	INPUT	
CLK	INPUT	E
NCLR	INPUT	
Q	OUTPUT	

Figure 5: *Port File*

quisition algorithm. The first is the name of the port, which is the 1^{st} token. The second is the type of the port, which is the 2_{nd} token. The third token is required for signals that influenced the result of function during state 0, state 1, rising and falling edge. A clock is an example signal that required this token. For function acquisition, the input vectors for combinational logic circuit is consisted of n^{th} series of binary 0 until n^{th} series of 1, where n is the total number of input ports obtainable from port file. For sequential logic circuit, the input vectors consisted of n^{th} series of binary 0 until n^{th} series of 1, and all possible combinations involving rising and falling signal edges. A full list of rising edge combinations ranges from $0r0...000$ to $1r1...111$, where r indicates a rise from 0 to 1. A full list of falling edge combinations ranges from $0f0...000$ to $1f1...111$. Each of these rise and fall combination is represented by two step cycle, `before` and `after` active signal edge. For example, a rising edge input vector combination $0r0...000$ is represented by `before` cycle 000...000, and `after` cycle 010...000. A falling edge input vector combination $0f0...000$ is represented by `before` cycle 010...000, and `after` cycle 000...000. Each bit in an input vector must correspond to each input port in the port information file. Figure 6 shows the main steps of input vector generation discussed above.

Recursive Cell Evaluation

The recursive cell group evaluation is the main body of DSPF function acquisition algorithm. This function served as the entry point where input vectors are assigned to ports and the assigned states are then propagated to transistor terminals. Evaluation of transistor to change the state of gate, drain and source will occur; changes in cycle property to 1 or 2 will occur depending on whether it is a first state or second state change. As discussed previously, the second state change happened during the second part of a rising or falling clock edge cycle. For rising edge, it is the period when active edge is 1. For falling edge, this is the period when active edge is 0. Evaluation is done to all transistors belonging to a cell, say a NAND gate, or a multiplexor before

e-ISSN: 2321-6980, ISBN (Print): 978-1-62776-963-1

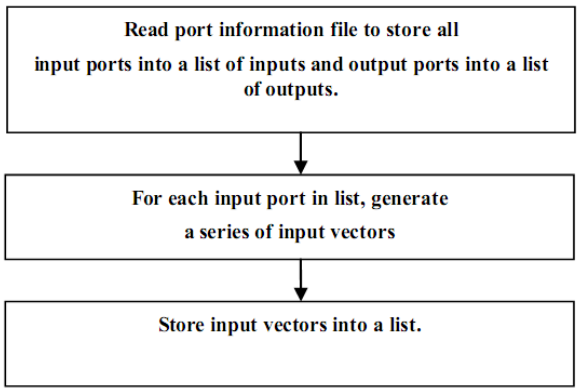

Figure 6: *Input vector generation*

evaluating transistors in other cells. After the evaluation of a cell, any state and cycle property changes to nets belonging to it will trigger a traversal to another cell. Identical transistor evaluation method and further traversal will occur with that cell. The evaluation of a cell will end once the net and transistor terminal states and cycle properties become stable, that is, no further changes occur. From the observation above, we can see that cell group evaluation works as a recursion. Each evaluation of transistor will be handled exclusively by another flow of the DSPF function acquisition algorithm. Figure 5 showed a detailed recursive cell group evaluation methology as described above. From figure 7, we can see that recursion will take place when state or cycle property changes are detected after comparing their values before and after evaluation of transistors. When this occurred, transistor terminals with changed states or cycle properties belonging to other cell provided a point of entry to these cells that need to be evaluated. When both state and cycle property are stable, no recursion will occur, and control is returned to parent, proceeding to evaluate other cells stored in list.

Evaluate Transistor

The recursive cell group evaluation algorithm takes in a transistor instance, determine it either as NMOS or PMOS, and then evaluate accordingly. State changes in source or drain will occur if the gate is opened and a signal is available in either the source or drain. So if a gate terminal is opened and data arrived at source terminal, that data will flow to the drain terminal. During the 1^{st} time a signal arrived at gate, source or drain, the cycle property of

e-ISSN: 2321-6980, ISBN (Print): 978-1-62776-963-1

that terminal is set to 1, indicating that this is before the rise or fall of active edge. During this 1^{st} part of the cycle, when the gate and either source or drain have cycle property of 1, the evaluation of third terminal is valid. During the 2^{nd} part of the cycle, however, an evaluation cannot occur when a single terminal is at cycle property of 2 while the second terminal is at cycle property of 1. The reason is because data during the 2^{nd} part of the cycle have not reached the second terminal, and evaluation of this transistor at this point in time is done using non stable second terminal state. In the event that a signal value stayed the same throughout the 1^{st} and 2^{nd} part of the cycle, the net and transistor terminal state will not change, but the cycle property need to be changed from 1 to 2, indicating that the signal has arrived and is valid during this 2nd cycle. When either source or drain state change occur after evaluation, the net connected to this terminal will need to be updated, both state and cycle property. This net is then traversed to obtain a list of transistor terminals belonging to the current cell. Figure 8 (a) and (b) showed the conditions when NMOS and PMOS transistor are evaluated. Figure 9 showed the flow during evaluation of transistors in a cell.

Conclusion

Evaluation of a digital circuit is completed once all states and cycle properties are stable for all cells. For combinational logic circuit, where rising or falling clock edge is not considered as affecting the output state, evaluation is completed at the end of cycle property 1. For sequential logic circuit, where rising or falling clock edge is affecting the output state, evaluation is completed at the end of cycle property 2. As explained previously, a rising or falling clock edge can be divided into 2 periods, `before` and `after` the active edge which corresponds to cycle property 1 and 2. Before changing to the next set of input vectors, the states and cycle properties need to be reset, identical to conditions described in Section 4. Figure 10 showed an example of D flip flop circuit with its function acquired as truth table in figure 11. In conclusion, this abstract has demonstrated an alternative methodology that is capable of performing function acquisition from customized standard cell. Further refinement could still be done to this approach, such as implementing Quinne-McCluskey logic minimization algorithm [5] to reduce the output into a single equation. Finally, the time complexity of this approach can be improved by multiprocessing, where each set of input vectors are analyzed by different machine.

References

[1] Lee, James M. Verilog Quickstart: A Practical Guide to Simulation and Synthesis in Verilog. Kluwer Academic Publishers, Norwell, MA, 1999.

[2] Tront, Joseph G. PSpice for Basic Circuit Analysis. McGraw Hill, New York, NY, 2007.

[3] Moore, Gordon E. Cramming more components onto integrated circuits. Electronic Magazine, 38, 8 (April 1965).

[4] Knuth, Donald E. Big Omicron and Big Omega and Big Theta. SIGACT News, 8, 18 (April-June, 1976).

[5] McCluskey, Edward J. Algebraic minimization and the design of two-terminal contact networks. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1956.

[6] Raj Gaurav Mishra and Amit Kumar Shrivastava, Implementation of Custom Precision Floating Point Arithmetic on FPGAs, HCTL Open International Journal of Technology Innovations and Research, Volume 1, January 2013, Pages 10-26, ISSN: 2321-1814, ISBN: 978-1-62776-012-6.

e-ISSN: 2321-6980, ISBN (Print): 978-1-62776-963-1

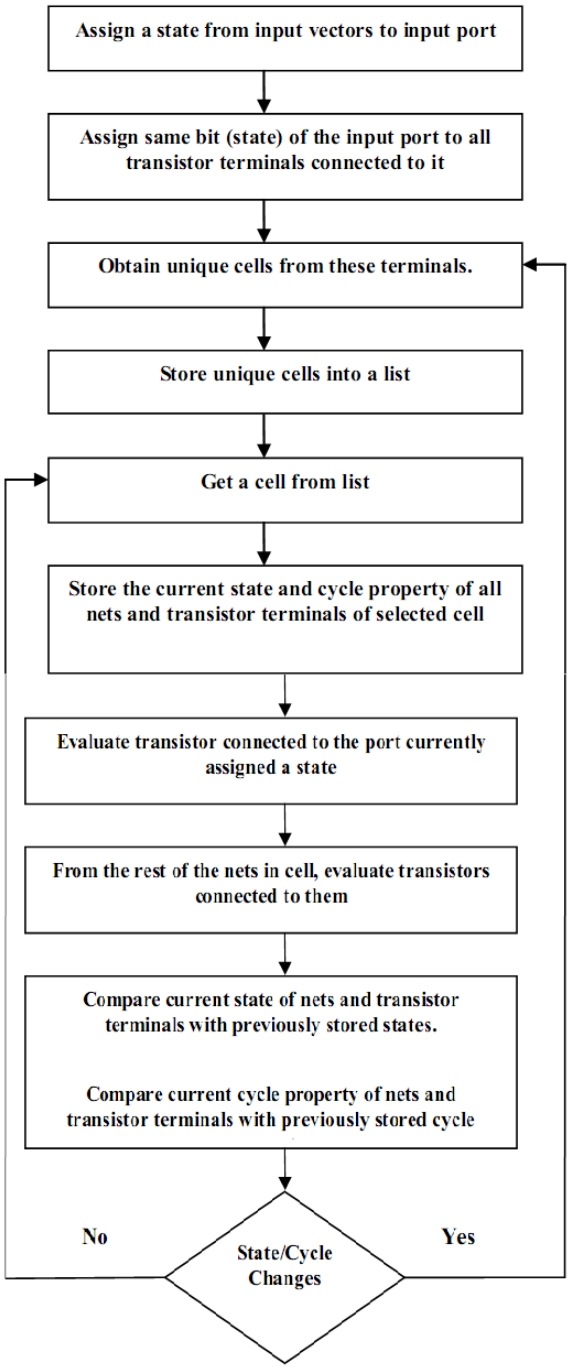

Figure 7: *Recursive cell evaluation*

e-ISSN: 2321-6980, ISBN (Print): 978-1-62776-963-1 *References*

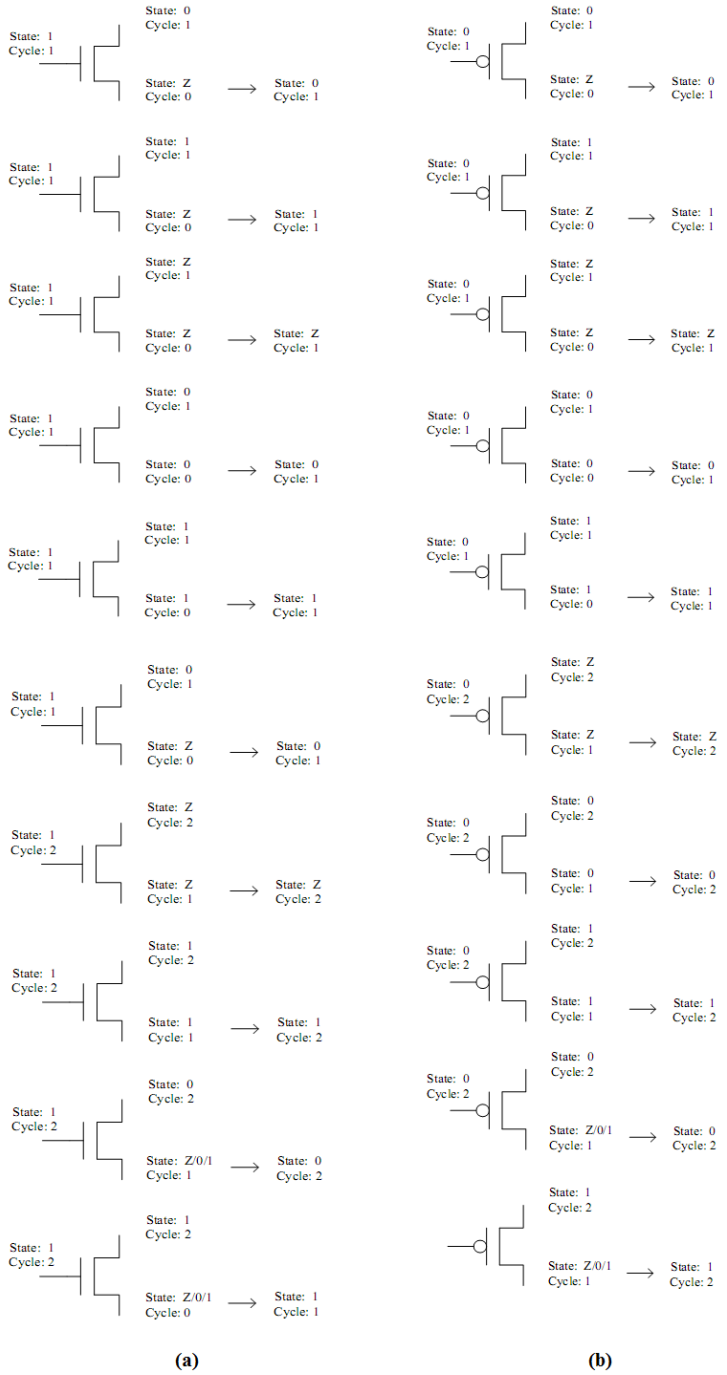

Figure 8: *(a) NMOS transistor evaluation, (b) PMOS transistor evaluation*

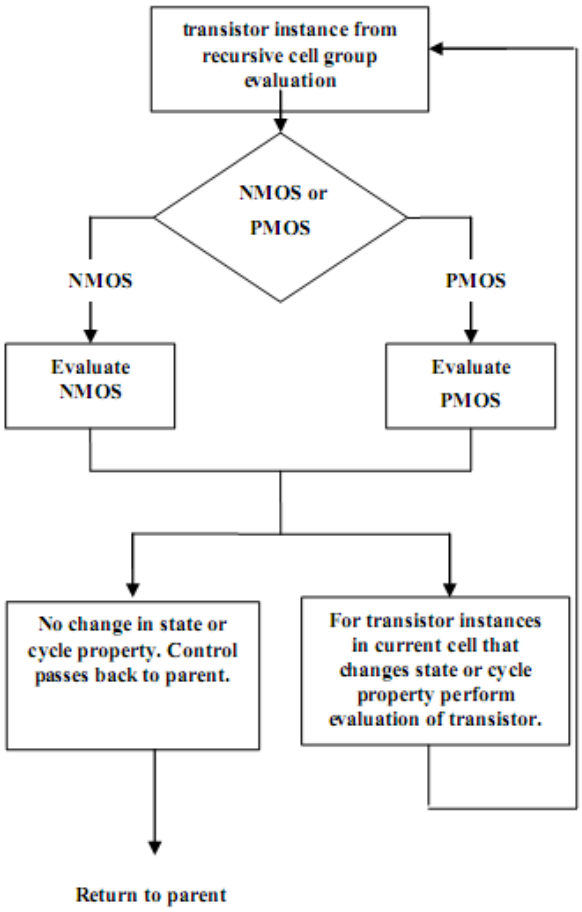

Figure 9: *Evaluation of transistors in a cell*

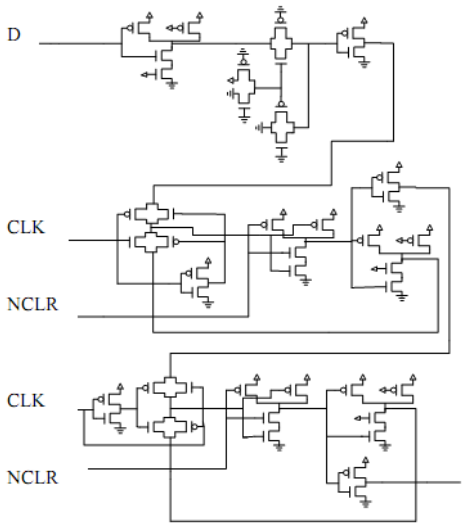

Figure 10: *Example D Flip Flop circuit*

```
                                                                  Q
D|CLK|NCLR +      Internal Node Signature                         +--
-----------+--------------------------------------------------+---
    000    | 10000010000110110110000100011001 | 0
    001    | 11000010100110111111000010101101 | Q
    010    | 10000010001010110010011100111001 | 0
    011    | 11000010111111111111011111111101 | Q
    100    | 10000001000110000111001100111010 | 0
    101    | 11000001110101001111101011101110 | Q
    110    | 10000001001010000011011100111010 | 0
    111    | 11000001111111001111111111111110 | Q
    0r1    | 11000010lrr11r111110rrrlr1r11101 | 0
    1r1    | 1100000111r1r10011111r1r111r1110 | 1
```

Figure 11: *Output from D Flip Flop*